John von Neumann Institute for Computing

Provision of Fault Tolerance with Grid-enabled and SLA-aware Resource Management Systems

F. Heine, M. Hovestadt, O. Kao, A. Keller

published in

Parallel Computing:
Current & Future Issues of High-End Computing,
Proceedings of the International Conference ParCo 2005,
G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, P. Tirado, E. Zapata (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **33**, ISBN 3-00-017352-8, pp. 113-120, 2006.

http://www.fz-juelich.de/nic-series/volume33

Provision of Fault Tolerance with Grid-enabled and SLA-aware Resource Management Systems*

Felix Heine[a] , Matthias Hovestadt[a], Odej Kao[a], Axel Keller[a]

[a]Paderborn Center for Parallel Computing (PC^2), Universität Paderborn, Germany

Future applications of the Next Generation Grid will demand for flexible negotiation mechanisms supporting various ways of Quality-of-Service (QoS) guarantees. In this context a Service Level Agreement (SLA) is a powerful instrument for describing all obligations and expectations within a business partnership. Many research projects already focus on realizing SLAs within the Grid middleware. However, this is not sufficient. Resource Management Systems also have to be SLA-aware, since these systems provide their resources to Grid infrastructures. In this paper we present the EU-funded project HPC4U (Highly Predictable Clusters for Internet Grids), which aims at realizing such an RMS by means of SLA-negotiation and SLA-aware scheduling, and application-transparent fault tolerance.

1. Introduction

Research on Grid computing started under solely technical aspects: how to realize this virtualization of resources, and how these distributed virtual resources can be used. Companies like IBM, Hewlett Packard and Microsoft have recognized the potential of Grid Computing and are investing noticeable efforts on research and the support of research communities. Common goal is to attract commercial users for Grid Computing. In this context, the European Commission convened a group of experts to clarify the demands of future Grid systems and which properties and capabilities are missing in currently existing Grid infrastructures. Their work resulted in the idea of the Next Generation Grid (NGG) [6].

The guaranteed provision of reliability, transparency and Quality of Service (QoS) is an important demand of the NGG. Commercial users will not use a Grid system for computing business critical jobs, if it is operating on the best-effort approach only. The user must be able to rely on getting the requested QoS level. At this, QoS does not only imply predictable operation of a single resource, but also the orchestrated execution of a workflow.

Service Level Agreements (SLAs) are powerful instruments for describing all expectations and obligations in the business relationship between service customer and service provider [2]. Such an SLA specifies the QoS requirement profile of a job. At the layer of Grid middleware many research activities already focus on integrating SLA functionality. However, there is a gap between the requirements of an SLA-aware Grid middleware and the capabilities of currently available Resource Management Systems (RMS), offering only best-effort service. Local resource management systems provide their resources to Grid systems, which transfer jobs from Grid users to these provided resources. Hence, SLAs guaranteed by the Grid middleware must be realized by local RMS [11]. However, if the local RMS operates at best-effort, Grid middleware can not assure specific service levels.

A major research focus at PC^2 is on resource management systems providing an increased level of quality of service as a software-only solution. At this, application transparency is crucial. Arbi-

*This work has been partially supported by the EU within the 6th Framework Programme under contract IST-511531 "Highly Predictable Cluster for Internet-Grids" (HPC4U).

trary applications should benefit without recompilation and linkage against special libraries from an increased level of QoS, like fault tolerant job execution. This is of particular interest for commercial Grid environments, since source code of commercial applications is normally not available.

Within the EU-funded project "Highly Predictable Cluster for Internet-Grids" (HPC4U) [7] the PC^2 is working on an SLA-aware resource management system, utilizing the mechanisms of the process, storage and network subsystems for realizing application-transparent fault tolerance. This paper will first describe the architecture of the HPC4U cluster middleware system, then explaining its mechanisms of QoS provision. The paper will conclude with a short summary and an overview about future work.

2. Related Work

The worldwide research in Grid computing resulted in numerous different Grid packages. Besides many commodity Grid systems, general purpose toolkits exist such as UNICORE [13] or Globus [5]. Although Globus represents the de-facto standard for Grid toolkits, all these systems have proprietary designs and interfaces. To ensure future interoperability of Grid systems as well as the opportunity to customize installations, the OGSA (Open Grid Services Architecture) working group within the GGF [3] aims to develop the architecture for an open Grid infrastructure [4].

In [6], important requirements for the Next Generation Grid (NGG) were described. Among those needs, one of the major goals is to support resource-sharing in virtual organizations all over the world. Thus attracting commercial users to use the Grid, to develop Grid enabled applications, and to offer their resources in the Grid. Mandatory prerequisites are flexibility, transparency, reliability, and the application of SLAs to guarantee a negotiated QoS level.

An architecture that supports the co-allocation of multiple resource types, such as processors and network bandwidth, was presented in [8]. The Globus Architecture for Reservation and Allocation (GARA) provides "wrapper" functions to enhance a local RMS not capable of supporting advance reservations with this functionality. This is an important step towards an integrated QoS aware resource management.

In this paper, this approach is enhanced by SLA and monitoring facilities. These enhancements are needed in order to guarantee the compliance with all accepted SLAs. This means, it has to be ensured that the system works as expected at any time, not only at the time a reservation is made. The GARA component of Globus currently does neither support the definition of SLAs or malleable reservations, nor does it support resilience mechanisms to handle resource outages or failures.

The requirements and procedures of a protocol for negotiating SLAs were described in SNAP [9]. However, the important issue of how to map, implement, and assure those SLAs during the whole lifetime of a request on the RMS layer remains to be solved. This issue is also addressed by the architecture presented in this paper.

The Grid community has identified the need for a standard for SLA description and negotiation. This led to the development of WS-Agreement/-Negotiation [1]. These upcoming standards rely on the new Web Services Resource Framework (WSRF, [10]) which will supersede the Open Grid Services Infrastructure (OGSI, [12]) specification. We will follow these developments closely and will stick to these standards.

3. Architecture of HPC4U

The HPC4U cluster middleware will consist of multiple elements, i. e. the SLA-aware RMS and the main building blocks for ensuring a high level of fault tolerance: process checkpoint, storage

snapshot and virtualization, and network failover. In an exceptional situation, e. g. the outage of hardware resources, the HPC4U system will use its fault tolerance mechanisms to assure the completion of a job. This means that the process checkpointing software enables checkpoint/restart (and migration) of a running process, so that jobs can be restarted from the last checkpoint on a spare resource. But only considering the checkpoint process could cause inconsistencies at restart, because a job continues to write data on files after it has been checkpointed. Hence, the system has to maintain consistency between the checkpointed process and the storage. The checkpointing process also has to be supported by the network subsystem, e.g. regarding in-transit network packets.

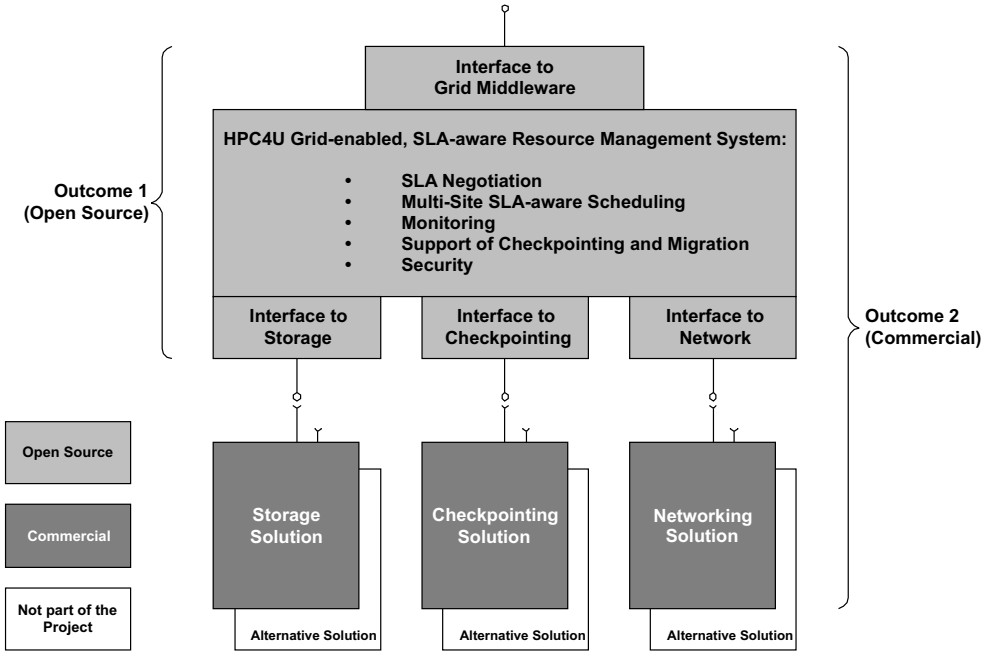

Figure 1. Outcomes of the HPC4U project

The results of HPC4U will be a mix of open source and proprietary software embedded in two outcomes (cf. Figure 1). The SLA-aware and Grid-enabled Resource Management System includes SLA negotiation, multi-site SLA-aware scheduling, security and interfaces for storage, checkpointing, and networking support. It will be multi-platform in nature and available as open source. The second HPC4U outcome will be a vertically integrated commercial product with proprietary Linux-specific developments for storage, networking and checkpointing. This outcome will demonstrate the entire, ready-to-use HPC4U functionality (job checkpointing, migration, and restart) for Grids based on Linux architectures. It is obvious that providing an agreed level of Quality of Service and Fault Tolerance requires broad interaction between all components of the HPC4U system.

Within the HPC4U project a cluster middleware system will be developed, which consists of three independent layers. At the upper level, HPC4U will provide an interface, which can be used by Grid middleware systems to negotiate on Service Level Agreements. To provide maximum flexibility and compatibility with other research projects and software systems, we will closely follow existing standards.

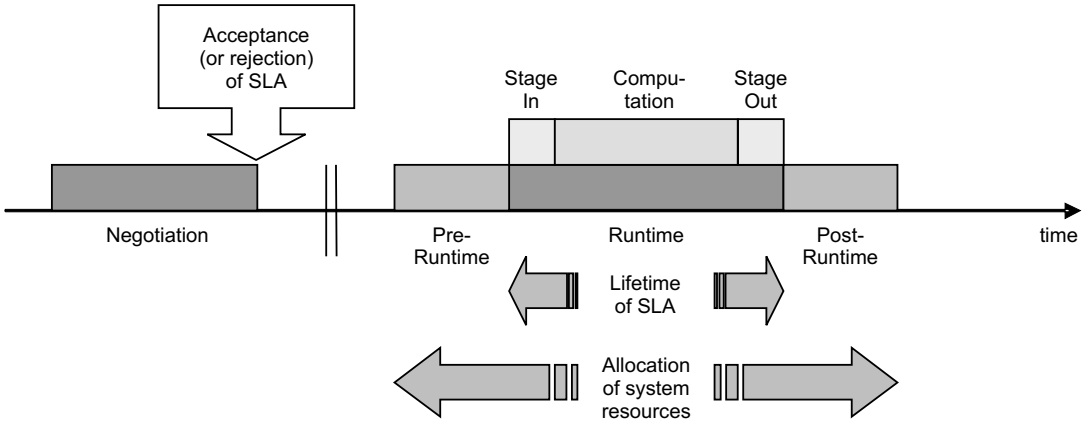

Figure 2. Phases of operation

An SLA-aware resource management system represents the middle layer of the HPC4U cluster middleware architecture. Using the above mentioned interface it negotiates with customers on SLAs. It also assures the compliance with these agreed SLAs at runtime. This does not only imply the monitoring of internal resources, but also the utilization of appropriate mechanisms to realize fault tolerance in case of resource outages.

These mechanisms base on functionalities which will be provided by the subsystems of HPC4U, which represent the lower level in the HPC4U cluster middleware architecture. Offering specific APIs, each of these subsystems provides special mechanisms for fault tolerance on process-, network- or storage-level. Since all interfaces within the HPC4U system are standardized and published, each component can be replaced with arbitrary third-party products, as long as these products provide compliant interfaces.

4. Provision of QoS

Negotiation is the initial step in the SLA lifetime. Here, the service consumer (e. g. end-user in the Grid) and service provider (e. g. an SLA-aware resource management system) first have to agree on the contents of an SLA. If the RMS agrees on the contents of an SLA request at the end of the negotiation process, it is responsible for fulfilling all demands and requirements of this SLA. Figure 2 depicts that the negotiation process may take place arbitrarily long in advance to the actual resource consumption. Between the successful completion of the negotiation process (which results in a new SLA) and the actual resource consumption starting at the pre-runtime phase, the RMS does not have to provide any resources for this SLA. However, it is aware of this SLAs, considering its demands in the scheduling process.

Before resources have to be provided according to the terms of the SLA, the system has to be configured (e. g. initialization of storage and network, or configuration of compute nodes) in the pre-runtime phase. These configured resources will be re-configured to normal operation after the job has been completed. During the runtime phase, the RMS has to ensure the adherence will all terms this SLA. For this, it uses its fault tolerance mechanisms for handling outages like failures of network, storage, or compute nodes.

4.1. Checkpointing of a Job

Resource outages (e. g. power failure of a compute node) usually cause a crash of the job running on these resources. Without checkpointing mechanisms, such a job has to be restarted from the very beginning, so that all computational results are lost. Especially for long running jobs, this is a major drawback. Deadline guarantees for such jobs are at most best effort. If a weather service uses Grid resources for computing the weekend weather forecast, the computed result would be useless if it is finished on Monday due to resource outages.

With system level checkpointing mechanisms available, the resource management system is able to create images of a running process in regular intervals. In case of resource outages, the resource management system can query for compatible and suitable spare resources to resume the job. If such resources were found, the checkpoint dataset (e. g. the process image) is transferred to the new compute node. There the job can restart from the last checkpoint. The impact on the time of completion of the job is minimal, since it is only delayed for the time required for checkpointing, dataset transfer, and restart. However, the job does not have to be restarted completely. This leverages the realization of fault tolerance and the provision of deadline guarantees. The HPC4U project will realize an application-transparent checkpointing, so that arbitrary applications can benefit from this service.

However, it is not sufficient to focus on process checkpointing only. The storage subsystem will provide checkpointing mechanisms for the storage partition of a running job. If the resource management system initiates a checkpoint of a running process, it simultaneously starts the checkpoint of the data partition. If the job needs to be restarted due to a resource failure at a later time, both process and storage are restored. This assures the consistency of the running process with its saved data.

If an application is running on multiple nodes in parallel, each node may send messages to the other nodes of the application (e. g. information exchange or synchronization). If such an application is checkpointed, a node might currently be sending a message to another node. At time of checkpoint, this message might have already left the sender, but not yet reached the recipient. Such a packed is called in-transit. Consistency is a major demand to the checkpointing mechanism. Hence, the network subsystem must handle these in-transit packets. At checkpoint time, the checkpointing subsystem will first freeze the application, so that its state is stable and does not change. Now the network subsystem is invoked to check all stacks for network packets. Also the network itself is checked for currently transmitted packets. All these packets are saved in a network dataset file.

4.2. Migration of a Job

To resume the computation of a job in case of a resource outage, the RMS first has to locate suitable free spare resources. For this query process, it can contact two information sources. First, the RMS knows about the SLA of the running job. This SLA already contains a detailed requirement profile of the job, which can be used for finding suitable resources.

But solely focusing on the SLA of a running application is not yet sufficient. The process subsystem of HPC4U will provide a mechanism called ”virtual bubble”. This bubble virtualizes the resources of a compute node. Applications which have been started inside such a bubble are only aware of these virtual resources, e. g. virtual network devices. By checkpointing the entire bubble, arbitrary applications can be checkpointed without any need for recompilation or relinking.

By checkpointing a running process within a virtual bubble, many dependencies between the application and its environment arise. These range from the type of the currently used processor, the operating system, and the exact kernel version, up to the location and exact version of libraries. The RMS needs also to cope with these constraints for finding really compatible spare resources. There-

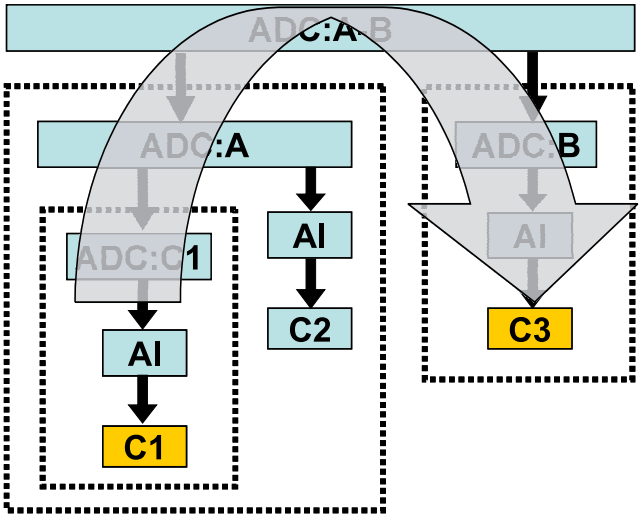

Figure 3. Migration to different cluster system

fore it first invokes the API of the checkpointing subsystem, querying for a compatibility profile of the checkpoint dataset. The checkpoint subsystem now starts to analyse the checkpoint dataset and returns and XML file, containing the demanded profile.

Based on these two sources of information, the RMS starts searching appropriate spare resources. At this, it tries to solve the problem as local as possible. This means, before migrating jobs to Grid resources, it first tries to find resources within the same cluster system or within the same administrative domain. Even if this strategy is not fixed and can be modified, it is preferable as the price of a migration process normally increases with the distance (i. e. the bandwidth capacity of the network between source system and target of migration).

Depending on the location of the found spare resource (local, administrative domain, Grid), the RMS invokes specific migration mechanisms. This implies the task of making the checkpoint dataset of the affected application available on the target resource.

If all preparatory tasks are completed, the RMS of the target resource invokes the API of the checkpointing subsystem to resume the job from the checkpoint dataset. The job will again run in a virtual bubble.

4.3. Example

In the HPC4U system, an Administrative Domain Controller (ADC) is responsible for establishing an administrative domain, where a given set of policies regarding security, access, accounting, or runtime responsibility is valid. The ADC serves as a central gateway between the internal resources of the administrative domain. Each of these internal resources (e. g. several cluster systems) is operated by means of a local resource management systems. To unify the interface between the ADC and the specific internal RMS, the ADC does not interface the RMS directly. Instead, an Active Interface (AI) acts as an adapter between the demands of the ADC and the capabilities of the specific internal RMS.

In the example scenario depicted in figure 3, cluster `C1` is operated by a subdepartment of a large company. This subdepartment has its specific policies on resource usage. Hence, the administrative domain controller `ADC:C1` is in charge of enforcing these policies. The subdepartment belongs to a department, again forming an administrative domain. This department is also operating cluster

system C2. The company also has a second department, operating cluster system C3. The company itself forms the surrounding administrative domain.

As explained above, cluster C1 is in charge of fulfilling all accepted SLAs. Assuming a resource outage at C1, C1 first tries to handle the problem internally, e. g. by rescheduling the jobs according the SLAs and resuming the job from the last checkpointed state. In case the RMS cannot realize the adherence with all SLAs (e. g. the resource outage has affected to much compute nodes), it tries to request for spare resources at the next higher level in hierarchy. ADC:A now tries to handle the problem internally, i. e. negotiating with cluster C2 on completing the job. If C2 is not able to accept the job (e. g. due to incompatible resources or high system utilization), ADC:A forwards the request to ADC:A-B, which is now trying to handle the problem internally. If cluster C3 accepts the SLA-request of C1, a migration process of all checkpoint data is initiated.

5. Conclusion and Future Work

In this paper we have outlined the basic ideas and components of the HPC4U cluster middleware system. HPC4U's main components are the SLA-aware resource management system and the subsystems for realizing fault tolerance on process, storage, and network.

The goal of the HPC4U project is to provide an application-transparent and software-only solution of a reliable Resource Management System. It will allow the Grid user to negotiate on Service Level Agreements, which will be realized by means of process and storage checkpointing, and other sophisticated mechanisms. By this, the HPC4U cluster middleware will be an important building block for realizing Next Generation Grids.

Application-transparency means that applications will not notice that they are running in an HPC4U environment, since they are running in the virtualized environment of a "virtual bubble". The application does not have to be modified, recompiled, or relinked in any way to benefit from the HPC4U fault tolerance mechanisms. Hence, the HPC4U system is able to provide checkpoint and migration service also for commercial applications where normally no source code is available.

The HPC4U solution will not only passively accept resource requests from Grid users, it will also act as an active Grid component. If the HPC4U system can not compensate resource outages, so that the fulfillment of agreed SLAs is endangered, it may request the Grid for suitable spare resources. If such resources are found, the job will be transparently migrated. This way, available Grid resources are used for further improving the level of Fault Tolerance.

Currently the HPC4U project is within the second of four technological workpackages. This workpackage addresses the first of three major steps in building up this system, namely the realization of the needed fault tolerance extensions to storage, communication, and system software in a single node environment. The development and implementation of these basic mechanisms serve as a fundament for merging single nodes into an Intranet Grid and then for including the Intranet Grid into the world wide Grids. The core tasks in this workpackage are related to preparing the building blocks for a grid-wide job migration and have the main goal, to integrate the job checkpointing with the storage and resource management component. Furthermore, a RMS monitoring mechanism will collect information about the available resources and their status.

The realization of this vertical approach is based on existing software solutions of the HPC4U partners. A first prototype implementation of our architecture has already been finished. It enables the user to request for a fault tolerant handling of his single-node jobs. The HPC4U system starts such a job within a virtual bubble, using the subsystems for transparent checkpoint and migration within the same cluster system. Ongoing work within HPC4U focuses on providing checkpointing and migration also to parallel-node jobs, and the realization of inter-cluster Grid migration.

Workpackage 2 will be finalized in mid 2005. Within the scope of the succeeding workpackage the existing FT mechanisms of the HPC4U system will be extended to multi-node/Intranet Grid environments on the one hand and to distributed running multi-node jobs on the other hand. The extension to Intranet Grids means, that the RMS must find suitable resources within this domain as a target for the job migration. This includes the suitability of the software and hardware architecture, the availability of the required resources and the compliance with the existing SLAs. Thus, for each migrated job, a start time for resumed processing will be assigned in a way that the given deadline can be reached, if the process duration information supplied by the user is correct.

The extension to multi-node jobs is a large scientific challenge, as already existing mechanisms are limited to single-node jobs. The migration of multi-node jobs affects the checkpointing, migration and restart mechanisms on job-, storage-, and communication-level, which must be able to deal with the specific characteristics of a multi-node job. The RMS has to be capable of handling multi-node jobs, since new requirements arise for compatibility, portability, and migration.

Moreover, the HPC4U system resulting of this workpackage will be capable of cross-border migration, allowing an RMS to migrate jobs on resources within the own administrative domain or over multiple administrative domains. This will further increase the Fault Tolerance, as (temporarily) HW/SW/Network failures can be compensated with a higher probability, as the pool of appropriate resources is significantly enlarged by considering all cross-border resources.

References

[1] A. Andrieux et al. Web Services Agreement Specification (WS-Agreement). `http://www.gridforum.org/Meetings/GGF11/Documents/draft-ggf-graap-agreement.pdf`, 2004.

[2] A. Sahai et al. Specifying and Monitoring Guarantees in Commercial Grids through SLA. Technical Report HPL-2002-324, Internet Systems and Storage Laboratory, HP Laboratories Palo Alto, November 2002.

[3] Global Grid Forum. `http://www.ggf.org`.

[4] GGF Open Grid Services Architecture Working Group (OGSA WG). Open Grid Services Architecture: A Roadmap, April 2003.

[5] Globus Alliance: Globus Toolkit. `http://www.globus.org`.

[6] H. Bal et al. Next Generation Grids 2: Requirements and Options for European Grids Research 2005-2010 and Beyond. `ftp://ftp.cordis.lu/pub/ist/docs/ngg2_eg_final.pdf`, 2004.

[7] Highly Predictable Cluster for Internet-Grids (HPC4U), EU-funded project IST-511531. `http://www.hpc4u.org`.

[8] I. Foster et al. A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation. In *7th International Workshop on Quality of Service (IWQoS), London, UK*, 1999.

[9] K. Czajkowski et al. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In U. Schwiegelshohn (Eds.) D.G. Feitelson, L. Rudolph, editor, *Job Scheduling Strategies for Parallel Processing, 8th InternationalWorkshop, Edinburgh,*, 2002.

[10] Karl Czajkowski et al. The WS-Resource Framework. `http://www.globus.org/wsrf/specs/ws-wsrf.pdf`, 2004.

[11] L.-O. Burchard et al. The Virtual Resource Manager: An Architecture for SLA-aware Resource Management. In *4th Intl. IEEE/ACM Intl. Symposium on Cluster Computing and the Grid (CCGrid) Chicago, USA*, 2004.

[12] S. Tuecke et al. Open Grid Services Infrastructure (OGSI) V1.0. `http://forge.gridforum.org/projects/ggf-editor/document/draft-ogsi-service-1/en/1`, 2003.

[13] UNICORE Forum e.V. `http://www.unicore.org`.